Claims

What is claimed is:

1. A synchronizable transactional database system comprising:

a bedrock layer for implementing a block-oriented storage abstraction with transactional support that allows for updates to be enacted atomically and to persist;

a bxtree layer for implementing a B+-tree on top of the bedrock that features data records of a form (key, value) and an additional operation to compute a digest of records within a range of key values, and;

an osynch layer for implementing a communication protocol for pairwise range synchronization of bxtree databases to simultaneously reduce local computation, bits communicated as well as communication rounds.

2. A synchronizable transactional database as in claim 1 wherein:

the bedrock layer uses a disk block array and a block mapping system to direct new data writes intended to overwrite a given logical block address to a free physical block leaving original data intact, and simultaneously the logical block address is tentatively mapped to the physical block containing the new data written.

3. A synchronizable transactional database as in claim 1 wherein:

the bedrock layer provides a commit operation that atomically switches all tentative maps to permanent maps for providing transactional capability at low level.

4. A synchronizable transactional database as in claim 1 wherein:

the bedrock layer operates within a single, preallocated file and does not use an outside logging facility.

5. A synchronizable transactional database as in claim 1 wherein:

the bedrock layer operates in a raw disk environment.

6. A synchronizable transactional database as in claim 1 wherein:

the bedrock layer writes to a shadow physical block until a commit or abort command updates a map array.

7. A synchronizable transactional database as in claim 1 wherein:

the bedrock layer consists of logical block addresses which are separated from physical blocks using a map array that stores correspondences between the blocks.

8. A synchronizable transactional database as in claim 7 wherein:

the bedrock layer mapping consists of a superblock level, a pages level and a map segments level which are kept in a main memory and assembled via a slalom procedure when an existing bedrock is open.

9. A synchronizable transactional database as in claim 1 wherein:

the bedrock layer consists of a slalom procedure abstracted and parameterized for read operations, write operations, and abort operations.

10. A synchronizable transactional database as in claim 1 wherein:

the bedrock layer supports parallel transactions.

11. A synchronizable transactional database as in claim 1 wherein:

the bedrock layer supports nested transactions.

12. A synchronizable transactional database as in claim 1 wherein:

the bedrock layer supports bedrock based file systems.

13. A synchronizable transactional database as in claim 1 wherein:

the bedrock layer supports in-block checksums.

14. A synchronizable transactional database as in claim 1 wherein:

the bedrock layer supports inter-block checksums for error correction.

15. A synchronizable transactional database as in claim 1 wherein:

the bedrock layer supports superblock redundancy.

16. A synchronizable transactional database as in claim 1 wherein:

the bedrock layer supports a bedrock block device controlling a new partition.

17. A synchronizable transactional database as in claim 1 wherein:

the bedrock layer supports memory caching.

18. A synchronizable transactional database as in claim 1 wherein:

the bedrock layer supports transparent in-memory block packing and unpacking.

19. A synchronizable transactional database as in claim 1 wherein:

the bxtree layer provides functions for insertion of records, deletion of records, retrieval of records, Get_All_Hashes and Get_Interval_Hashes.

20. A synchronizable transactional database as in claim 1 wherein:

the bxtree layer supports variable length data records and keys and data is written to it in a portable fashion.

21. A synchronizable transactional database as in claim 1 wherein:

the digest of records of the bxtree layer is compacted by a cryptographically strong function.

22. A synchronizable transactional database as in claim 1 wherein:

internal nodes of the bxtree layer form an index over leaves of the tree where the data resides;

leaf nodes of the bxtree store a fixed size digest for each record which is used to verify the records integrity and used by functions providing support for range synchronization, and;

the internal nodes store a set of keys to guide a search for records.

23. A synchronizable transactional database as in claim 1 wherein:

the bxtree layer stores for each internal node a triplet of the form (address, num_records, hash) where address is the address of a child node, num_records is the number of records in a child's sub-tree, and hash is the XOR of the digest of the records in the child's sub-tree.

24. A synchronizable transactional database as in claim 1 wherein:

each node of the bxtree layer is stored in a separate bedrock block, and;

each bedrock block has the same number of bytes of storage, whereby,

full leaf nodes can have different numbers of records and full internal nodes can have different numbers of keys.

25. A synchronizable transactional database as in claim 1 wherein:

the records of the bxtree layer are of the form (key, version, value) where the version field can be used by handler functions that are invoked by the osynch layer.

26. A synchronizable transactional database as in claim 1 wherein:

the osynch layer invokes a Get_Interval_Hashes function to compute a single summary of all records lying in a given key interval creating a local summary and a remote summary;

the remote summary is transferred to a local database and compared with the local summary for matching, and;

the size of the remote database restricted to the given key interval is checked for the number of records lying in the key interval whereby: if the number of records is larger than a predetermined number the remote database partitions the key interval into smaller sub-intervals and transfers summaries for each sub-interval to the local database wherein each sub-interval is then synchronized.

27. A method of synchronizing values of a database system comprising:

providing for updates to be enacted atomically and to persist for implementing a block-oriented storage abstraction with transactional support in a bedrock layer;

implementing a B+-tree on top of the bedrock that features data records of a form (key, value) and an additional operation to compute a digest of records within a range of key values as a bxtree layer, and;

implementing a communication protocol for pairwise range synchronization of bxtree databases to simultaneously reduce local computation, bits communicated as well as communication rounds as an osynch layer.